TP n° 5 : Analyse de données en pandas

Objectifs du TP : Manipuler des bases des données avec Pandas, affichage graphique avec Matplotlib.

- Données de pollution sur Paris (Source : Airparif) -

Commençons par charger les données `20080421_20160927-PA13_auto.csv`. Si besoin, vous aurez peut-être à installer le package `download` avec la commande suivante :

```
pip install download # à exécuter dans une cellule si besoin.
```

Sinon, vous pourrez aussi charger les données à la main, et passer la partie téléchargement automatique de la cellule suivante.

Téléchargement automatisé des données :

```
# Download
import numpy as np
import pandas as pd
from download import download
url = "http://josephsalmon.eu/enseignement/datasets/20080421_20160927-PA13_auto.csv"
path_target = "./20080421_20160927-PA13_auto.csv"
download(url, path_target, replace=False)
```

Ceci étant fait, les données sont disponibles dans votre répertoire (le vérifier avec `ls` par exemple).

On va maintenant importer ces données avec `pandas` en utilisant la commande suivante :

```
pollution_df = pd.read_csv('20080421_20160927-PA13_auto.csv', sep=';',
                           comment='#', na_values="n/d",
                           converters={'heure': str})
pollution_df.head(25)
```

1) Regarder la colonne des heures et commenter. Observer ce que produit la commande suivante pour remédier au problèmes rencontré :

```
# 24:00 issues, voir https://www.tutorialspoint.com/python/time_strptime.htm
# pollution_df['heure'] = pollution_df['heure'].replace('24', '0')
pollution_df['heure'] = pollution_df['heure'].astype(int) - 1
pollution_df['heure'] = pollution_df['heure'].astype(str)
```

2) Maintenant que l'on a normalisé les heures on va les importer dans un format de type `datetime` de `pandas` pour faire des manipulations simples sur les jours, mois, années, etc. Considérez les diverses étapes qui suivent et décrivez ce que l'on vient de faire :

```
time_improved = pd.to_datetime(pollution_df['date'] +
                               ' ' + pollution_df['heure'] + ':00',
                               format='%d/%m/%Y %H:%M')

pollution_df['DateTime'] = time_improved
del pollution_df['heure']
del pollution_df['date']
```

```
pollution_ts = pollution_df.set_index(['DateTime'])
pollution_ts = pollution_ts.sort_index()

# Seulement les 4 années pleines
day_ini = '01/01/2009'
day_end = '12/31/2015'
pollution_ts = pollution_ts.loc[day_ini:day_end]
pollution_ts.head()
```

3) Passer en revue la base de données avec la fonction `describe()` de `pandas`. Que vous inspire le fait que l'attribut `count` soit différent pour les deux gaz ?

4) Afficher en utilisant la commande `resample` sur le dataframe `pollution_ts` l'évolution de la concentration moyenne par jour sur toute la durée d'étude. On utilisera pour cela début :

```
%matplotlib notebook
import matplotlib.pyplot as plt
import seaborn as sns
fig, axes = plt.subplots(2, 1, figsize=(9, 4), sharex=True)
names_polluant = ['NO2', 'O3']
for i, polluant in enumerate(names_polluant):
    axes[i].plot() # XXX TODO
```

5) Afficher l'évolution de la pollution journalière pour les deux polluants (NO2 et O3) : cette fois on cherche à obtenir le profils hebdomadaire de la pollution : on utilisera la commande `resample` pour afficher la moyenne journalière pour les 7 jours de la semaine.

6) La pollution atmosphérique montre-t-elle une tendance à la baisse au fil des ans ? Pour cela on pourra faire une visualisation simple des moyennes annuelles sur la période d'étude.

7) Afficher le profils par mois sous forme de graphique en barres : on affichera pour les douze mois de l'année autant de barres qu'il y a d'années complètes. On cherche donc à produire le graphique suivant [1] :

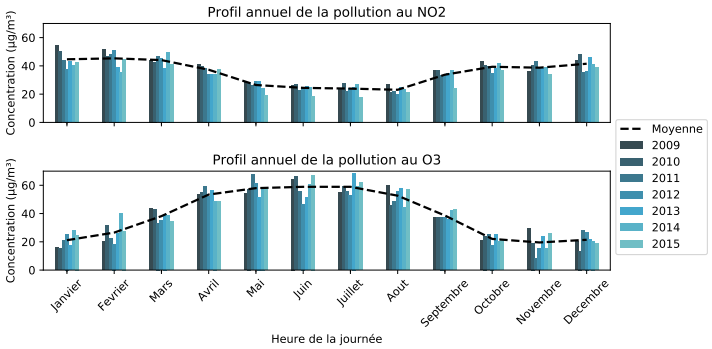

FIGURE 1 – Pollution : profils par mois

8) Au vue des seuils légaux, voir par exemple https://www.airparif.asso.fr/reglementation/normes-europeennes, trouvez combien de fois les valeurs limites et les seuils d'alertes ont été franchi sur la période d'étude.

9) Trouver les 10 pics de pollutions les plus importants pour les deux polluants (jour et heure). Commentez.

- DONNÉES DE CONSOMMATION ÉLECTRIQUE (SOURCE : EDF) -

1. on pourra utiliser la fonction `groupby` pour cela, *cf.* https://jakevdp.github.io/PythonDataScienceHandbook/03.08-aggregation-and-grouping.html

On utilise la base de données [2] **Individual household electric power consumption Data Set**. Pour cela utiliser les commandes ci-dessous :

```
# download part if needed.
from download import download
import os
import zipfile

# url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00235/'
# Lien alternatif:
url='http://josephsalmon.eu/enseignement/datasets/household_power_consumption.zip'

filename = 'household_power_consumption.zip'
cwd = os.getcwd()
path_target = os.path.join(cwd, filename)
download(url, path_target, replace=False)

# # unzip part
zip = zipfile.ZipFile(path_target)
zip.extractall()

# Visualisation succinte du fichier décompressé:
!head -10 'household_power_consumption.txt'
```

10) Quelle est la nature des colonnes dans cette base données ? Dans la suite et sauf contre-indication on n'utilisera que la variable `Global_active_power`.

11) Charger la base de données avec `read_csv` :

```
na_values = ['?', '']
fields = ['Date', 'Time', 'Global_active_power']
df_conso = pd.read_csv('household_power_consumption' + '.txt', sep=';',
                       na_values=na_values, usecols=fields)
```

12) Selon vous, quel est le symbole qui encode les données manquantes ici ?

13) En utilisant `describe` donner le nombre de ligne de la base de données et la moyenne de la variable `Global_active_power` sur toutes la durée de l'étude.

14) Utiliser `df_conso.tail()` et `df_conso.head()` pour trouver les dates de début et de fin d'étude.

15) En utilisant `df_conso.count()` calculer le pourcentage de lignes manquantes dans la base de données `df_conso`.

16) Lancer et décrivez ce que fait l'instruction suivante :

```
df_conso = df_conso.dropna(axis=0)
print('Taille da la base sans valeurs manquantes: {} lignes'.format(
      df_conso.shape[0]))
```

17) Utiliser `to_datetime` et `set_index` pour créer un objet `Serie` indexé par le temps (on prendra garde au format des dates internationales qui diffère du format français, et on utilisera l'option `infer_datetime_format=True` pour accélérer). On supprimera ensuite les colonnes `Date` et `Time` qui ne serviront plus, et on ne gardera que les années complètes.

18) Afficher le graphique des moyennes journalières entre le 1er janvier et le 30 avril 2007. Proposer une cause expliquant l'évolution de la consommation fin février et début avril.

19) Reprendre le question précédentes mais cette fois en produisant quatre sous-graphiques (avec `plt.subplot`) pour représenter les unes en dessous des autres les quatre années complètes de la base de données.

2. Voir https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption pour un descriptif des données

20) Proposer une visualisation pour analyser le comportement par jour de la semaine et mettre en évidence les différences de comportement entre le weekend et le reste de la semaine.

21) Faire de même pour analyser le comportement par mois au sein du foyer.

- Ajouter des données climatique -

On ajoute des informations de température pour cette étude : les données utiles étant disponibles ici http://josephsalmon.eu/enseignement/TELECOM/MDI720/datasets/TG_STAID011249.txt [3]. Ici les températures relevées sont celles d'Orly (noter cependant qu'on ne connaît pas le lieux de relevé de la précédente base de données).

22) Charger les données avec `pandas`, et ne garder que les colonnes `DATE` et `TG`. Diviser par 10 la colonne `TG` pour obtenir des températures en degrés Celsius. Traiter les éléments de température aberrantes comme des NaN.

23) Créer une `Serie pandas` des températures journalières entre le 1er janvier et le 30 avril 2007. Afficher sur un même graphique ces températures et la séries `Global_active_power`.

3. on peut aussi trouver d'autres informations sur le site http://eca.knmi.nl/dailydata/predefinedseries.php